漢字構造情報のIPLD化の試み

AUTHOR(S):

守岡, 知彦

CITATION:

守岡, 知彦. 漢字構造情報のIPLD化の試み. 東洋学へのコンピュータ利用 第30回研究セミナー 2019: 1-15

ISSUE DATE:

2019-03-08

URL:

http://hdl.handle.net/2433/241263

漢字構造情報の IPLD 化の試み

守岡 知彦

1　はじめに

CHISE では（その前身である UTF-2000 の頃から）『Chaon モデル』と呼ぶ方法によって文字を表現するようになっている。これは汎用符号化文字集合に依存することなく自由に文字を表現するために我々が提案しているもので、表現したい文字に関する知識（文字の性質の集合）の機械可読な表現によって文字を表現し操作する方法である。Chaon モデルでは、文字を説明するための要素（文字の性質や用例など）を『文字素性』(character feature) と呼ぶ。文字素性としては、部首、画数、部品の組合せ方に関する情報（漢字構造情報)、発音、意味、用例、その他文字処理で必要となる各種情報などが考えられる（図 1)。

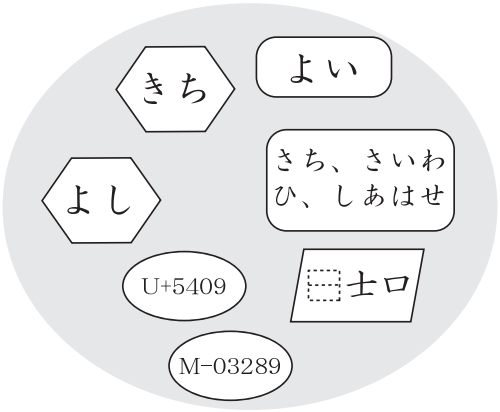

図 1　Chaon モデルにおける文字オブジェクトの概念図

CHISE では Chaon モデルに基づき文字素性の集合による文字定義を行い、それに基づいた文字処理の仕組みや文字情報サービスを提供してきた。この CHISE の仕組みは汎用符号化文字集合に対してその性質や他のソースとの対応関係等を記述するためのメタシステムを提供するものといえ、文字の符号化作業や外字の整理等において有用なツールを提供することができたが、Chaon モデルの本来の理念という観点から見れば、現行の実装 (XEmacs CHISE) は文字の指示において ID 素性を必要とするという点で問題があったといえる。

XEmacs CHISE における文字処理、あるいは、Chaon モデルに基づく文字処理は文字の性質を文字オントロジーと呼ぶデータベースにその基礎を置いているといえるが、ID 素性 というのはリ

レーショナルデータベースにおける主キーに相当するものといえる。XEmacs CHISE では ID 素性のない文字定義を使って文字オブジェクトを定義することも可能であるが、この場合、全く同じ文字定義を用いたとしても別の文字オブジェクトとして作成されてしまう。[*1] この問題を避けるためには最低でも 1 つの ID 素性を文字定義に含める必要がある。

Web の世界で CHISE の文字情報をやりとりするためには文字オブジェクトやその構成要素に IRI を割り振ることが必要といえ、この観点では全てのオブジェクトに ID 素性を付与すべきだといえる。しかしながら、恣意的に付与せざるを得ない ID 素性を要することは分散環境において自由に文字を表現し交換するという目標に対して重大な制約を科すものといえる。また、CHISE 文字オントロジーの基盤であるグラフストレージの Concord では ID 素性の値としてシンボルや文字列や整数も利用可能であるが、CHISE では歴史的経緯から非負整数のみが利用可能である。

この制約が実際に問題となっている対象の一つが漢字構造情報の処理である。IDS を用いた漢字構造記述では IDS を入れ子状に記述することができるが、この文字になっていない中間部品はそのままでは ID 素性を持たないため問題を引き起こす訳である。原理的には、もし文字オブジェクトでも ID 素性の値として文字列やシンボルを用いることができれば、GlyphWiki が行っているように IDS に基づく ID を用いることができるといえるが、CHISE では部品として UCS の抽象文字以外のものも利用可能であり、また、多粒度包摂モデルに基づいた部品の包摂粒度を記述する必要があるため、IDS を ID として用いるだけでは解決できないといえる。[*2]

しかしながら、もし文字定義そのものを ID のように使えれば ID 素性無しに同じ（構造等価）な文字定義は常に同じ文字オブジェクトを指すことになると考えられる。例えば、文字定義に何らかの正規化を施しその（暗号学的）ハッシュ値を用いることが考えられる。ここでは P2P ベースの分散型ファイルシステムの一つである IPFS とそのデータモデルである IPLD を用いて実際に漢字構造情報を表現することを試みる。

2 IPFS

IPFS (InterPlanetary File System) [1] [4] は Protocol Labs 社が中心となって開発しているオープンソースの P2P 型分散型ファイルシステムである。IPFS の開発は Git 等で版管理された科学的データを高速に転送するシステムの構築を目的として Juan Benet によって始められたが、後に、分散化され永続化された Web として構想されるようになった。

IPFS では Git と同様にオブジェクトをそのハッシュ値によって参照するという内容に基づくアドレッシング (content-addressing) を用いている。通常の Web では URL (IRI) という資源の場所に基づくアドレッシング (location-addressing) になっており、識別子の指す先が変化する可能性があり、また、オブジェクトに変化がなくてもその識別子が変化してしまうと参照できないという性質を持っている。このことは学術資源の長期保存においては良い性質とはいえず、この

[*1] Lisp におけるリストが (eq '(1 2 3) '(1 2 3)) → nil となるのと同様である。つまり、文字定義の同一性は構造等価を意味してもオブジェクトの同一性は意味しない訳である。
[*2] GlyphWiki でも IDS で指示される文字のバリエーションを示すために恣意的な ID を使わざるを得ないといえる。

ため、DOI のような永続的識別子の利用が注目されているといえる。この観点から IPFS を見ると、IPFS では DOI のような永続的識別子を任意のオブジェクトに対して機械的に生成する仕組みと見ることができる。IPFS ではオブジェクトのハッシュ値に基づいて内容 ID (CID; Content IDentifier) を生成する仕組みがプロトコルによって定義されており、いつどこで誰が行っても対象とするオブジェクトが同一のバイト列を持っていれば同じ CID が生成されるため、DOI や DNS のように機関を認証して権限を委譲する必要はなく、結果的に、非常に小さな粒度のデータから非常に大きなデータセットの集合といったスケーラビリティーで永続的な識別子を生成することができる。

IPFS は

アプリケーション	
IPNS	名前解決
IPLD	Merkle DAG によるデータモデル
libp2p	ネットワークやルーチング、データ転送

という 4 層で構成されており、下位の 3 層及び IPFS が提供する幾つかのアプリケーション[*3]に関して、JavaScript と Go 言語による実装が提供されている。[*4]

2.1 IPLD

IPLD (Inter Planetary Linked Data) [5] [3] は IPFS のデータモデルを提供する層で、Merkle DAG と呼ばれる暗号化されたハッシュに基づく有向非巡回グラフ (Directed Acyclic Graph; DAG) と、Merkle paths と呼ばれる Merkle DAG を名前付きリンクをたどってトラバースしたものを UNIX 風の階層的ファイル名に対応させる仕組みと、IPLD の正規化されたデータ形式 (IPLD Canonical Format) 等の仕様により、JSON や XML 等の外部形式を IPLD の世界に格納したり IPLD の世界のオブジェクトを指定した外部形式で出力することができる。

IPLD は IPFS のデータモデルであるので、IPFS 上のさまざまなアプリケーションは全て IPLD の DAG として記述されたデータを持っているといえるが、ここでは IPFS の Go 実装 (go-ipfs) の ipfs コマンドの ipfs dag {get|put} や IPFS HTTP API の /api/v0/dag/{get|put} で読み書きできる IPLD dag-cbor 形式を用いるものとする。

IPLD のオブジェクト（DAG のノードとなるもの）は JSON のオブジェクトと同様な key と value の対の集合である。

例えば、

{ "name": "Paul Marie Ghislain Otlet" }

は `zdpuAwbrw9r5jy1gUoB61EoNLGRossy171TGKyB2AbAvmArmo` という CID を持つ IPLD のオ

[*3] 分散ファイルシステムとしての IPFS もこのアプリケーション層に位置するものと看做すことができる。
[*4] go-ipfs と js-ipfs はこのプロトコルスタックの実装をパッケージにまとめたもののようである。

ブジェクトの JSON 表現である。

IPLD に JSON のオブジェクトを格納する場合、IPFS のようにバイト列そのものが格納される訳ではなく、正規化された CBOR (Concise Binary Object Representation) [2] 形式に変換されて格納される。

例えば、go-ipfs の ipfs コマンドを用いて、

```
% echo '{ "a": 1, "b": 2, "c": 3 }' | ipfs dag put
```

を実行すると、

```
zdpuAmNMQJdQumv32j5DDUGBzk92aZmXcg2x78Tqrj3pPV2Ze
```

という CID を得るが、

```
% echo '{ "c": 3, "a": 1, "b": 2 }' | ipfs dag put
```

や

```
% echo '{"a":1,"b":2,"c":3}' | ipfs dag put
```

でも

```
zdpuAmNMQJdQumv32j5DDUGBzk92aZmXcg2x78Tqrj3pPV2Ze
```

という CID になり、空白・改行の有無・個数や key-value 対の順番に関らず同じオブジェクトとして扱われることが判る。

また、現状、go-ipfs の dag サブコマンドでは JSON 以外の形式がサポートされていないが、原理的には、XML や YAML, RDF といった形式であっても、それが同一の key-value 対の集合を表現しているならば同一のオブジェクトとして扱われることになっている。

IPLD では他のオブジェクトへのリンクは key が `"/"` で value が CID である link-object と呼ばれる特殊なオブジェクトで表現される。例えば、

```
{ "/": "zdpuAwbrw9r5jy1gUoB61EoNLGRossy171TGKyB2AbAvmArmo" }
```

は link-object の JSON 表現である。また、

```
{ "name":  "Henri La Fontaine",
  "collaborator":
    { "/": "zdpuAwbrw9r5jy1gUoB61EoNLGRossy171TGKyB2AbAvmArmo" }
}
```

を JSON 表現として持つ IPLD オブジェクト

```
zdpuAoDrX6sbMNE5N3i7YTrVuEidCL41EoSryCr5y8UhrR4PC
```

のようにある key の value に他のオブジェクトへのリンクやリンクの配列を入れることができ、これにより DAG を構成することができる。

Merkle-paths の規定により、リンクを持つオブジェクトはリンクを値として持つ key を用いて、CID/key のような派生的 CID によってリンク先を参照することができる。

例えば、

% `ipfs dag get zdpuAoDrX6sbMNE5N3i7YTrVuEidCL41EoSryCr5y8UhrR4PC/collaborator`

は {`"name":"Paul Marie Ghislain Otlet"`} という出力を返す。

もし、リンク先にも同様に名前付きのリンクがある場合、CID/a/b/c/... のように階層的にリンクをたどることができる。

3 IPLD での表現上の注意点

IPLD では JSON で記述されたデータのハッシュ値に基づいて内容 ID (CID) を生成するため、構造等価な JSON データは同じ CID になる。CHISE 的にいえば、同じ素性対の集合で記述された（即ち、同じ文字定義を持つ）2 つの文字オブジェクトは必ず同一（一意に定まる）ということである。言い替えれば、ある文字オブジェクトに文字素性を追加したり素性値を書き換えたら CID が変化し、別の文字オブジェクトになってしまうということである。

こうしたことから、CHISE の文字データ[*5]を IPLD 化する場合、変化しにくい部分と変化しやすい部分に分けて扱う方が良いと考えられる。また、変更履歴や各版の CID を捜し出す仕組みが必要であろう。後者は Git 等の分散型版管理システムが行っているようなことを行うことだといえる。あるいは、これに加えて逆リンクを辿るための仕組みが必要なこともあるかも知れない。

文字オブジェクトに含まれる情報の内、文字符号や字書等での文字番号に関するもの、即ち、従来 ID 素性で表現してきたような情報は不変性が高いといる。しかしながら、複数の ID 素性対を同一視して束ねる場合、どれとどれを束ねるかには恣意性があったり新たな ID 素性が追加され得る（個々の ID 素性対には不変性があったとしても）ことに注意する必要がある。こうした場合、不変性の高い幾つかの CID が同一であることを IPLD の外側[*6]で管理する必要があるだろう。変更しやすい部分の管理や逆リンクも同様にこうした不変性の高い CID をキーにした名前解決の仕組み（データベース）によって実現することができると考えられる。[10]

4 符号化文字の IPLD での表現

漢字構造をデータ化する場合、そのノードとなる部品をどのように表現するかが問題となる。即ち、IPLD の世界において CHISE の文字オブジェクトをどのように表現するかを考える必要があ

[*5] これはおそらく構造データ一般でも成り立つだろう

[*6] IPNS や IPLD 上での分散台帳（仮想通貨）を使うなどして IPLD を基盤として使うことも考えられるが、この場合も IPLD とは違うレイヤーで扱っている訳で、ここでは IPLD の外側と考える。

るといえる。

CHISE の文字オブジェクトを IPLD 化する場合、その文字定義情報をそのまま IPLD の JSON に直訳することも可能であるが、前節で述べた議論や RDF との親和性を考慮し、CHISE の RDF 化 [9] において行ったモデルを踏まえて設計することにする。

[9] では ID 素性が担っている機能を

1. 文字符号の符号位置を示す
2. 包摂粒度を示す
3. 複数の ID 素性対を持つ文字オブジェクトにおいて、それらの素性対の等価性（指し示すものが同じであること）を示す

という 3 種類に分析し、

符号位置 IRI　ある文字符号におけるある符号位置を示す IRI. 文字符号を示す IRI と符号位置から構成される。

包摂粒度を示す述語　文字オブジェクトに対応する IRI（これを『文字 IRI』と呼ぶことにする）と符号位置 IRI を関係づける（表 1）。

文字 IRI の等価性の記述　異なる CCS の符号位置 IRI に対応した文字 IRI が存在する時にその等価性を `owl:sameAs` によって示すことで実現できる。但し、当面、文字定義の Turtle 文書では `owl:sameAs` と等価な述語 `:eq` を用いることにする。

の 3 種類の記述に分けて扱うことにした。

包摂粒度名	S 式	EgT[7]	RDF 述語
超抽象文字	`==>`	`a2`	:super-abstract-character-of
抽象文字	`=>`	`a`	:abstract-character-of
統合字体	`=+>`	`o`	:unified-glyph-of
抽象字体	`=`	`rep`	:abstract-glyph-of
詳細字体	`=>>`	`g`	:detailed-glyph-of
抽象字形	`==`	`g2`	:abstract-glyph-form-of
字形	`===`	`repi`	:glyph-image-of

表 1　包摂粒度の表現

今回の CHISE の文字オブジェクト の IPLD 化ではこの枠組に基づき、

1. 符号位置オブジェクト
2. 包摂情報付き符号位置オブジェクト
3. 符号化文字オブジェクト

の 3 種類に分けて考える。また、今回は漢字構造記述に限定して考えるため、ID 素性以外の一般

の文字素性は考えないことにする。[*7]

また、階層的素性名が使われている場合、ベースとなる素性名に対応した述語がとる目的語として空白ノードを設け、その空白ノードの述語 `:context` で階層的素性名におけるドメイン情報を表現する。また、述語 `:target` で素性値を表現することにしていた（図 2）が、IPLD 化においても同様の表現を行うことにする。

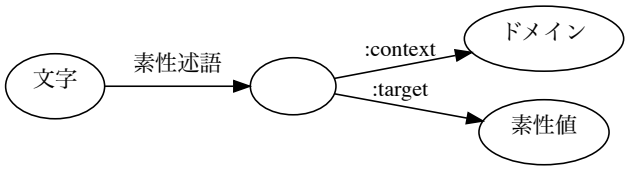

図 2　ドメイン付き素性名の表現

4.1　符号位置オブジェクト

符号位置オブジェクトは

{ 文字符号名：符号位置 }

のような文字符号名と符号位置の対で表現する。但し、文字符号名は CHISE の ID 素性名から包摂粒度を示す接頭辞（表 1）や階層的素性名でのドメインを取り除いたものである。また、符号位置は整数で表現する。[*8]

例えば、素性対

```
(=>ucs@iwds-1 . #x5167)
```

に対応する符号位置オブジェクトは

{ ”ucs”：20839 }

という JSON で表現でき、その CID は

`zdpuAxm2netKNpcrd29ToLk9hsvrvEXpAgLDEAGsW6AU4zaD1`

となる。

[*7] 一般の文字素性を扱う場合、符号化文字オブジェクトに紐づける形で文字オブジェクトを定義すれば良いと考えられる。

[*8] 10 進数で記述する。

4.2 包摂情報付き符号位置オブジェクト

包摂情報付き符号位置オブジェクトは符号位置オブジェクトと包摂粒度を示す述語を用いて、

{ 包摂粒度を示す述語：
{ ”code-point”：{ ”/”：符号位置オブジェクトの CID } }
}

のように表現する。但し、包摂粒度を示す述語は表 1 に示す RDF での包摂述語と同じものを用いる（接頭辞は取る）。

例えば、素性対

```
(=>jis-x0208 . #x4662)
```

に対応する包摂情報付き符号位置オブジェクトは符号位置オブジェクト

{ ”jis-x0208”：18018 }

の CID

```
zdpuAovC2r7BJj5hZVuMKmWZh5U8xnHtgArgZHoRfXqW2vUnz
```

を用いて、

{ ”abstract-character-of”：
{ ”code-point”：
{ ”/”：”zdpuAovC2r7BJj5hZVuMKmWZh5U8xnHtgArgZHoRfXqW2vUnz” }
}
}

という JSON で表現でき、その CID は

```
zdpuAtw9b6Qbmna3DkzvkXYuJWQ6LqSuQTCL9RiySFd4Ajorc
```

となる。

また、階層的素性名が使われている場合、

{ 包摂粒度を示す述語：
{ ”code-point”：{ ”/”：符号位置オブジェクトの CID } },
{ ”context” ：{ ”/”：ドメインオブジェクトの CID } }
}

のように表現する。ここで、ドメインオブジェクトは

{ "CHISE-domain" : ドメイン名 }

のように表現する。

例えば、`=>ucs@iwds-1` のドメイン `@iwds-1` は

{ "CHISE-domain" : "iwds-1" }

という JSON で表現でき、その CID は

`zdpuB2dY1V16SxFjEM9PpDUHwLyRWTafDtaB4AB5QmiJ7yFVv`

となる。そして、

`(=>ucs@iwds-1 . #x5167)`

に対応する包摂情報付き符号位置オブジェクトは符号位置オブジェクト

{ "ucs" : 20839 }

の CID である `zdpuAxm2netKNpcrd29ToLk9hsvrvEXpAgLDEAGsW6AU4zaD1` を用いて、

```
{ "abstract-character-of" :
    { "code-point" :
        { "/" : "zdpuAxm2netKNpcrd29ToLk9hsvrvEXpAgLDEAGsW6AU4zaD1" }
    },
    { "context" :
        { "/" : "zdpuB2dY1V16SxFjEM9PpDUHwLyRWTafDtaB4AB5QmiJ7yFVv" }
    }
}
```

という JSON で表現でき、その CID は

`zdpuB21qbgNRx4aNssEL3Ji6sBApvugiVU9ekfZ6RbzkXWC91`

となる。

4.3 符号化文字オブジェクトの IPLD での表現

符号化文字オブジェクトは同じ文字を指し示す包摂情報付き符号位置オブジェクトの CID の集合 $\{C_1, C_2, C_3, ...\}$ を用いて

{ "unify" : [{ "/" : C_1 }, { "/" : C_2 }, { "/" : C_3 }, ...] }

のように表現する。

例えば、文字素性対の集合

```
((=>ucs@iwds-1 . #x5167)
 (=>ucs@jis . #x5185)
 (=ucs@big5 . #x5167)
 (=>jis-x0208 . #x4662)
 (=>jis-x0213-1 . #x4662)
 (=big5 . #xA4BA)
 (=>iwds-1 .  0148))
```

に対応する符号化文字オブジェクトは

```
{
  "unify": [
    { "/" : "zdpuAqoV6gKUXhF3TS5yaHtWpBa8rySqeqooKpfrwP2YCkoCy" },
    { "/" : "zdpuAmaApUWKEgcLpuN4w8hdykB4THmfw9fFwx8mXzSBriFo5" },
    { "/" : "zdpuB21qbgNRx4aNssEL3Ji6sBApvugiVU9ekfZ6RbzkXWC91" },
    { "/" : "zdpuAtw9b6Qbmna3DkzvkXYuJWQ6LqSuQTCL9RiySFd4Ajorc" },
    { "/" : "zdpuAxug4bQU3hNJ2Jp4AC5bDYTm4QXo4mtLDDF69D3xWdQpK" },
    { "/" : "zdpuAnhJaMBoRgTTLvKyFRd53Cv4g6axLm4iQazbGwms4ejph" },
    { "/" : "zdpuAyxTYkacukYnFfk6qmukXeStGjfJSWZhfX2PbWkFXimuo" }
  ]
}
```

という JSON で表現でき、その CID は

```
zdpuAwsV7iswdPXqPH7HGHAeQSgqL4fxmMXrbz3o1tr3ovcYH
```

となる。ここで、

```
zdpuAqoV6gKUXhF3TS5yaHtWpBa8rySqeqooKpfrwP2YCkoCy は (=big5          . #xA4BA),
zdpuAmaApUWKEgcLpuN4w8hdykB4THmfw9fFwx8mXzSBriFo5 は (=ucs@big5     . #x5167),
zdpuB21qbgNRx4aNssEL3Ji6sBApvugiVU9ekfZ6RbzkXWC91 は (=>ucs@iwds-1  . #x5167),
zdpuAtw9b6Qbmna3DkzvkXYuJWQ6LqSuQTCL9RiySFd4Ajorc は (=>jis-x0208   . #x4662),
zdpuAxug4bQU3hNJ2Jp4AC5bDYTm4QXo4mtLDDF69D3xWdQpK は (=>ucs@jis     . #x5185),
zdpuAnhJaMBoRgTTLvKyFRd53Cv4g6axLm4iQazbGwms4ejph は (=>iwds-1       . 0148),
zdpuAyxTYkacukYnFfk6qmukXeStGjfJSWZhfX2PbWkFXimuo は (=>jis-x0213-1 . #x4662)
```

の CID である。

5　漢字構造情報の IPLD での表現

CHISE 文字オントロジーでは漢字構造情報は IDS 形式をパースした結果の構文木を S 式にしたものとして表現しており、その文字素性として ideographic-structure を用いている。

CHISE の漢字構造情報を IPLD 化する場合、その ideographic-structure 素性をそのまま IPLD の JSON に直訳することは可能であるが、RDF との親和性を考慮し、CHISE の RDF 化 [9] で行ったのと同様な『IDC の述語化』や IDS 用コンテナを使ったモデル [8] を用いることにした。

[9] と異なるのは、IDC を `"operator"` というメンバーに格納するということと、包摂粒度を示す述語の場合と同様、IDC に対応する述語も接頭辞は取るということである。

例えば、「字」の漢字構造は

```
{
  ”operator” : ”\u2FF1”,
  ”above” : { ”/” : "zdpuArp21drHhyzxKZDXSsJ4G1QvXNsGgpUK6evUq7aqatNJv" },
  ”below” : { ”/” : "zdpuArnniMgYWGyhxnQNhwzDi4FR4148CT8prAoyvnkwKwwzS" },
}
```

という JSON で表現でき、その CID は

```
zdpuB2d2yHpE3EdgBjgyF371mZktmknqXmyqJYMNMYBMmjXgL
```

となる。

また、U+5B57 の抽象文字を示す包摂情報付き符号位置オブジェクトの漢字構造情報は

```
{
  ”operator” : ”\u2FF1”,
  ”above” : { ”/” : "zdpuB2qk4Bna1D3QE3Mb8WMf7yRdZ3dcSNkxN2Te9QJBZzWCV" },
  ”below” : { ”/” : "zdpuAxHGREWVdV6Skj8DnHP6z5aH85ajLAufqAhgYLukT1f5N" },
}
```

という JSON で表現でき、その CID は

```
zdpuAmm22JYYkdPxBsdzBku2zCBWkZrkPZ2hLTbxQFPNzaTdX
```

となる。

6　Concord の利用

IPFS/IPLD ではデータの内容（データのハッシュ値）に基づく ID (CID) によってオブジェクトを管理しているため、データの内容を 1 バイトでも書き換えると別の ID になってしまう。言い

替えれば、IPLD のオブジェクトは不変なオブジェクト (immutable object) であり、オブジェクトの編集とは編集前のオブジェクトを編集後のオブジェクトに置き換える行為であるといえる。そして、編集前のオブジェクトと編集後のオブジェクトの CID は別のものであるので、両者を関係づける仕組みが必要であるといえる。

また、ものが存在する場所で示される (location-addressed) もの（例えば、大漢和辞典の文字番号 12345 の文字や URL で示される Web 頁、あるいは、Git リポジトリの指定したバージョンや指定したブランチの最新版など）やその性質で示されるもの（例えば、人の年齢や今日の湿度など）といった形で対象物を指示したい場合もある。

あるいは、あるオブジェクト A が別のオブジェクト B へのリンクを持つ場合、オブジェクト B からオブジェクト A への逆リンクを記録したいとする。もし、この逆リンクをオブジェクト B に記述するとオブジェクト B の CID が変化してしまい、オブジェクト A から参照しているものとは別物になってしまう。そこで、オブジェクト A からのリンクを新しいオブジェクト B に置き換えると今度はオブジェクト A の CID も変化してしまう。それ故に、DAG 構造しか記述できない訳であるが、現実問題として、逆リンクの情報を記述するとしたらそれはオブジェクト B の外側で行うしかないといえる。

IPFS には IPNS という名前解決のための仕組みが存在する。これは公開鍵暗号技術を用いて IPFS ネットワークのノードに Peer ID というものを割り当て、電子署名技術を用いて Peer ID に IPFS の CID を対応づけるものである。よって、これは編集され得るサイトの最新版を配信するような用途には使えるものの、IPLD のさまざまなオブジェクトをその場所や性質で指示するための仕組みとしてはそのままでは使えない。また、Peer ID に紐づけられるという性質から特定のノードを利用可能な利用者しか登録できず、分権化 (decentralize) という観点では問題である。[*9]

きちんと分権化された仕組みに基づいてこうした問題を解決することが重要であるといえるが、当面のアドホックな解決法として既存の Web 技術を併用することで location-addressing の実現や逆リンクや付加情報の管理を行うことを試みた。これは具体的には IPLD のオブジェクトに対応する Concord [6] オブジェクトを作成し、EST [7] (CHISE-wiki) を用いて参照するという方法である。

6.1 符号位置オブジェクト

符号位置オブジェクトに対しては `code-point` ジャンルの Concord オブジェクトを生成する。

Concord における符号位置オブジェクトはその ID として IPLD の CID を用い、ID 素性 `=ipld` にも CID を格納する。

また、IPLD のオブジェクトのメンバーはその名前の前に `=` を付けた ID 素性を用いて表現する。但し、この `=` は Concord において ID 素性にするために付けた接頭辞であり、包摂粒度を示すものではない。

[*9] きちんとやるためには、分散台帳を実現する必要があると思われる。

6.2 包摂情報付き符号位置オブジェクト

包摂情報付き符号位置オブジェクトに対しては `coded-character` ジャンルの Concord オブジェクトを生成する。

Concord における包摂情報付き符号位置オブジェクトはその ID として IPLD の CID を用い、ID 素性 `=ipld` にも CID を格納する。

包摂粒度を示す述語に対応する素性は IPLD のメンバー名の接尾辞 `-of` を取り、*-of* を示す関係素性の接頭辞 `<-` を付けたものを用いる。これにより対応する符号位置オブジェクトに逆関係素性が自動的に付与される。

また、ドメイン付きの情報の場合、ドメイン付きの階層的素性名を用い、空白ノードに相当するオブジェクトは作らない。

また、利便性のために、対応する CHISE の文字オブジェクトを `character` 素性に格納している。

6.3 符号化文字オブジェクト

符号化文字オブジェクトに対しても、包摂情報付き符号位置オブジェクトと同様に、`coded-character` ジャンルの Concord オブジェクトを生成する。

包摂情報付き符号位置オブジェクトと同様に、Concord における符号化文字オブジェクトはその ID として IPLD の CID を用い、ID 素性 `=ipld` にも CID を格納する。

IPLD におけるメンバー `unify` は関係素性 `->unify` で表現する。これによりその素性値に含まれる各包摂情報付き符号位置オブジェクトに符号化文字オブジェクトへの逆関係素性 `->unify` が自動的に付与される。もし、符号化文字オブジェクトが再定義されれば、新たな符号化文字オブジェクトへの逆関係素性が蓄積されることになる。

また、利便性のために、包摂情報付き符号位置オブジェクトと同様に、対応する CHISE の文字オブジェクトを `character` 素性に格納している。

6.4 漢字構造オブジェクト

漢字構造情報を表現する IPLD オブジェクトに対しては `glyph` ジャンルの Concord オブジェクト（漢字構造オブジェクト）を生成する。

実用上、`coded-character` ジャンルを用いても良いが、漢字構造記述には漢字の抽象形状を示すことを目的にしたものと文字がどういう音符・意符・形符の組み合わせから成り立っているかを示すことを目的にしたもの（解字）があり、両者を区別するために、現状の漢字構造情報は `glyph` ジャンルに入れ、将来的に解字情報を `coded-character` ジャンルに入れることを検討している。ただ、この問題に関しては本稿では詳述しない。

Concord における漢字構造オブジェクトはその ID として IPLD の CID を用い、ID 素性

=ipld にも CID を格納する。

IDC を示すメンバー operator は同名の素性 operator で表現する。

IDC に対応する述語を示すメンバー名は接頭辞 -> を付けた関係素性で表現する。これにより部品オブジェクトに逆関係素性が自動的に付与される。

また、対応する coded-character ジャンルのオブジェクトに対して漢字構造オブジェクトへのリンクとなる関係素性 ->ideographic-structure を付与する。これにより漢字構造オブジェクトから coded-character ジャンルのオブジェクトへの逆関係素性が自動的に付与される。この結果、もし、複数のオブジェクトが同じ漢字構造を持つ場合、その集合が漢字構造オブジェクトの素性 <-ideographic-structure に集積することになる。

7 おわりに

CHISE の漢字構造情報の IPLD 化について述べた。本稿で述べた手法により IPLD 上で漢字構造情報を表現することができ、また、IPLD に基づく表現を用いることによって文字ではなく IDS で記述された部品の同一性を保証することができた。

IPLD ではデータの書き換えによってオブジェクトの内容 ID (CID) が変化してしまうため、変化しないオブジェクトを手がかりに変化したオブジェクトを探すための仕組みが必要である。また、逆リンクの実現にも同様の仕組みが必要であるといえる。本稿では Concord をこのための仕組みとして用いたが、IPFS の名前解決のための枠組である IPNS を用いる方法を検討することも重要な課題といえる。また、CHISE 全体の IPLD 化も今後行うべき重要な課題といえる。

参考文献

[1] Juan Benet. IPFS - content addressed, versioned, P2P File System (draft 3). *arXiv preprint arXiv:1407.3561*, 2014 年.

[2] Carsten Bormann and Paul Hoffman. *Concise Binary Object Representation (CBOR)*. Internet Engineering Task Force (IETF), 2013 年 10 月. RFC 7049.

[3] David Dias. IPLD—the “thin-waist” merkle dag format. https://github.com/ipld/specs/blob/master/IPLD.md.

[4] Protocol Labs. IPFS is the distributed web. https://ipfs.io/.

[5] Protocol Labs. IPLD. https://ipld.io/.

[6] 守岡知彦. Concord: プロトタイプ方式のオブジェクト指向データベースの試み. Linux Conference 抄録集, Vol. 4, , 2006 年.

[7] 守岡知彦. Wiki 的手法に基づく構造化データの編集について. 人文科学とコンピュータシンポジウム論文集 —人文工学の可能性〜異分野融合による「実質化」の方法〜, 情報処理学会シンポジウムシリーズ, 第 2010 巻, pp. 33–40. 情報処理学会, 情報処理学会, 2010 年 12 月.

[8] 守岡知彦. 漢字構造情報の RDF 化の試み. 山崎直樹（編）, すべてをコンピュータの中に—

繋がってしまったデータとその未来, pp. 3–22. 京都大学人文科学研究所共同研究プロジェクト: 情報処理技術は漢字文献からどのような情報を抽出できるか—人文情報学の基礎を築く, 全国共同利用・共同研究拠点「人文学諸領域の複合的共同研究国際拠点」, 2013 年 2 月.

[9] 守岡知彦. CHISE の RDF 化の試み. 情処研報, Vol. 2017-CH-114, No. 1, pp. 1–6, 2017 年 5 月.

[10] 守岡知彦. 古典中国語 UD コーパスの IPFS を用いた表現の試み. 情処研報, Vol. 2018-CH-118, No. 6, pp. 1–7, 2018 年 8 月.